Manufacturing Interoperability

S.R. Ray & A.T. Jones

National Institute of Standards and Technology, Gaithersburg, MD 20899, USA

ABSTRACT: As manufacturing and commerce become ever more global in nature, companies are increasingly dependent upon the efficient and effective exchange of information with their partners, wherever they may be. Leading manufacturers rely upon computers to perform this information exchange, which must therefore be encoded for electronic transmission. Because no single company can dictate that all its partners use the same software, standards for how the information is represented become critical for cost-effective, error-free transmission of data. This paper discusses some interoperability issues related to current standards, and describes two projects underway at the National Institute of Standards and Technology in the areas of interoperability testing, and in self-integration research. We believe that tomorrow's standards will rely heavily upon the use of formal logic representations, and that these will enable automation of many integration tasks.

Corresponding author:
Dr. Steven Ray
NIST
100 Bureau Drive, M/S 8260
Gaithersburg, MD 20899-8260

Email: ray@nist.gov
Phone: +1 301 975 3524
Fax: +1 301 258 9749

1 INTRODUCTION

The term “manufacturing interoperability” refers to the ability to share technical and business information seamlessly throughout an extended manufacturing enterprise (supply chain). This information, previously shared in a variety of ways including paper and telephone conversations, must now be passed electronically and error-free with suppliers and customers around the world. Disparate corporate and national cultures, a plethora of international and regional standards, and numerous commercial products make this task all the more difficult, further underscoring the need for a clear and unambiguous interoperability infrastructure. The penalty paid by industry for the lack of such an infrastructure has been quantified in a 1999 study commissioned by NIST (NIST, 1999). This study reported that the U.S. automotive sector alone expends one billion dollars per year to resolve interoperability problems. The study also reported that as much as 50% of this expenditure is attributed to dealing with data file exchange issues.

There are three principal approaches used to reduce these exorbitant costs. The first is a point-to-point customized solution, which can be achieved by contracting the services of systems integrators. This approach is expensive in the long run because each pair of systems needs a dedicated solution. When there are, for example, ten partners in the chain, this would require up to 90 (10x9) interfaces. Moreover, should any system provider release a software upgrade, many of the translators would likely need modification.

A second approach, adopted in some large supply chains, is for each original equipment manufacture (OEM) to mandate that all supply chain partners conform to a particular, proprietary solution. This has been the practice, for example, in the automotive sector. While this may solve the interoperability problem for the OEM, it does not solve the interoperability problem for the partners. This happens because the first or sub-tier suppliers are forced to purchase and maintain multiple, redundant systems if they want to do business with several major OEMs.

The third approach involves neutral, open, published standards. By adopting open standards the combinatorial problems associated with interoperability becomes of order N rather than order N^2 as described above. When there are ten partners, only ten bi-directional translators are needed. Published standards also offer some stability in representation of information, an essential property for long-term data archiving. This long-term data retention issue is increasingly recognized as a costly yet critical problem, particularly for industries with long product life cycles, such as aerospace. One example of a successful open standard is ISO 10303 (ISO 1994a), informally known as STEP, the STandard for the Exchange of Product Model data. STEP, which is actually a family of standards, defines a neutral representation for product data over its entire life cycle. The most widely adopted component, ISO 10303-203 (ISO 1994b), (Configuration controlled design) is already conservatively estimated to be saving the transportation equipment manufacturing community over $150 million per year in mitigation and avoidance costs, with the figure expected to rise to $700 million by 2010 (NIST, 2002).

But the problem is far from solved. Interoperability standards are used in layers, from the cables and connectors, to the networking standards, to the application or content standards such as STEP. All of these layers must function correctly for interoperability to be achieved. Figure 1 shows an example of the array of choices that a manufacturer must face to be able to conduct electronic commerce today. The greatest challenges remain at the top of this stack, vertical content interchange standards.

Figure 1 - Stack of Interoperability Standards

2 IS XML THE ANSWER?

Many data interchange standards groups are adopting XML (the eXtensible Markup Language, available at http://www.w3.org/XML/) as the basis for specifying their data content standards. XML has had a tremendously positive impact on the connectivity of systems, but also has more clearly exposed what problems remain. XML is a markup language that can be used to tag collections of data with labels. As part of a standardization activity, communities can agree on the names for these labels. An interoperability problem remains, though, if different people have differing

understandings of the meaning of an XML tag. Stated more succinctly, XML standardizes the syntax of data exchange, but was never designed to capture the semantics of the data. This is not necessarily an obstacle for a tightly knit community that operates within a common context. In this situation, the mental associations with a tag are shared and well understood by all. Where this limitation becomes a problem is in moving data from one context to another, for example sending data from a manufacturing context to a financial context. Without explicit, rigorous definitions of terms, misunderstanding is sure to arise.

Researchers at NIST, recognizing that we need a better way to capture definitions of terms, initiated the development of the Process Specification Language (PSL) (Gruninger, 2003). Upon searching for the best conceivable way to capture definitions of terms, the team adopted the answer suggested by the philosophy community (which, after all, has been pondering this question for a number of centuries). That community advocated the use of first-order logic, which brings the ability to reason over sets of definitions and to prove properties of these sets. One such property, for example, is consistency, which is tedious and prone to error using traditional information-modeling techniques. Using logic, it becomes straightforward to ensure the consistency of assertions for large sets of definitions by using automated theorem provers.

PSL, which is logic based, ensures rigorously defined and consistent definitions for data sets. This is a worthwhile goal in its own right, but even more exciting capabilities follow. Once a software system is equipped with a logic-based set of definitions (often called an ontology), then that system can advertise its outputs and desired inputs in a

manner that can be manipulated and “understood” by other systems. This begins to move systems beyond the “screen scraping” techniques that are sometimes used today to collect data. Finally, such a rigorous foundation for data definitions can be the basis for reaching the holy grail of systems integration – self-integration.

In this paper we describe in more detail two projects that are part of our efforts to develop techniques that may lead to self-integration: the Automated Methods for Integrating Systems (AMIS) project, and the B2B (Business-to-business) Interoperability Testbed.

3 AUTOMATED METHODS FOR INTEGRATING SYSTEMS (AMIS)

An integration problem is typically stated as a requirement to produce an improved business result from a set of software components (see Figure 2). Each component that contributes information or functionality to the improved business result exposes interfaces. These interfaces are the communication vehicles by which the individual components will interact and thereby form a new integrated system. The process of integration is the activity that joins the components appropriately, with modification if necessary, so that that the components together may accomplish new functions. Functions accomplished by such interactions are called joint actions.

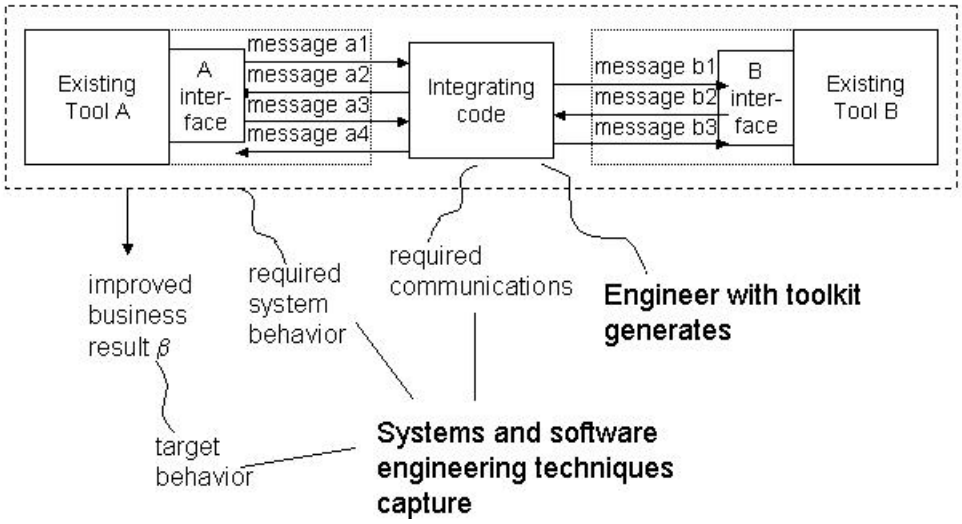

Figure 2 - Generic View of the Integration Process

As shown in Figure 2, tool A and tool B each expose interfaces for communication in a number of possible forms including, messages, sharable files, or middleware interfaces. A systems engineer reasons from the desired business result to determine the target feature or behavior the new system must yield – the joint action the two systems are to perform. From the required system behavior and the exposed interfaces, the systems engineer determines the required communications between tool A and tool B.

To enable the communications, the engineer generates the specifications to translate the relevant messages between the two tools using the interfaces they support and forms that are meaningful. From these specifications, software developers can generate integrating code translators to produce the necessary translations and obtain the new system behavior.

The research question is: can we automate the building of translators that work directly from the interface specifications, as human engineers do? To answer this question, the AMIS project has embarked on three main areas of work: capturing the business and engineering interactions in models that are in a form suitable for machine reasoning, building tools to implement semantic links between models, and, generating translations between the tools based on those semantic links.

3.1 *Model Formulation*

Systems engineers perform the integration task using a combination of top-down and bottom-up approaches to match business process objectives with component functionality. Our approach is to encode the information into formal models that (1) allow software-based reasoning tools to automate some the integration tasks, and (2) assist the systems engineers in performing the remaining tasks. Three such models have been identified: the joint action model, the local interaction model, and the engineered interface model.

The joint action model (JAM) is a requirements model for the intended joint action that will produce the desired business result. The JAM is abstracted from the relevant concepts in the envisioned business interaction. It contains the required interactions between the component systems with roles for each and a shared model of the business entities pertinent to those interactions.

A local interaction model (LIM) is a conceptual model that captures the business entities that are significant to the functions of the component, along with their properties and

relationships, each of the "business activities" (functions) performed by the component, and any business activity or function it expects from some other component, along with the business entities and properties it uses or alters. Figure 3 shows the relationship between a LIM and the JAM.

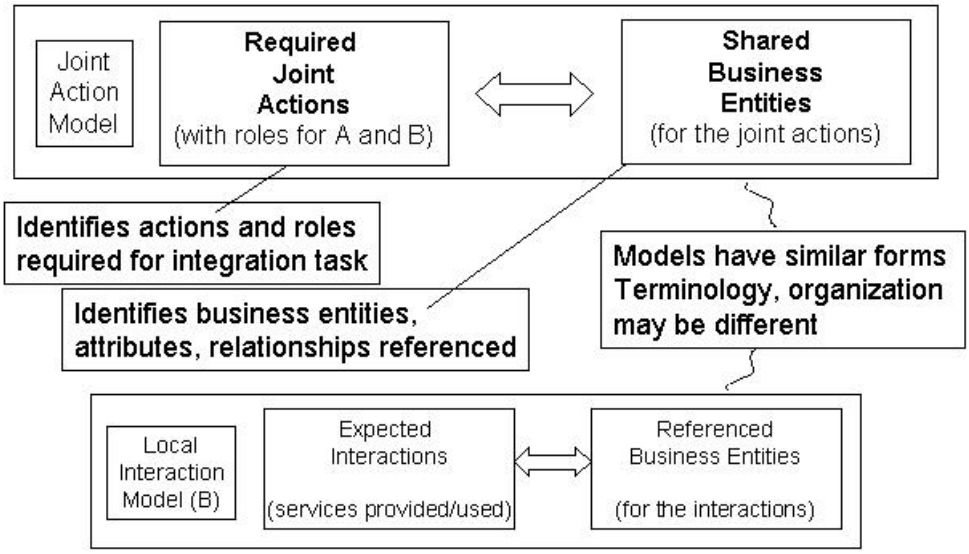

Figure 3 - Relationship Between the JAM and the LIM

Each LIM is generated from the engineered interface model (EIM), which captures the interfaces supporting possible interactions with other systems or agents. Methods of communicating the information contained in those interfaces include file transfer, database access, operation invocation, or queued messaging (as shown in Figure 2). For each unit of communication there is a mechanism, and each agent plays a specific role with respect to that mechanism. And for each unit of communication, there is a message, which is the set of information transferred by that communication unit. A further level of detail, which defines the detailed protocols for the communications and the binary representations for each data item, is also included.

Although conceptual models and engineering models serve different roles, they are not used in isolation. Inter-model relationships between elements from these views link the relationships between an activity or entity expressed in business terms and an engineering means of implementing that activity or representing the entity.

3.2 *Semantic Mapping*

After the LIMs and the JAM have been produced, the relevant notions in the LIMs must be semantically mapped to their corresponding notions in the JAM. The AMIS approach is to build tools once to automate the creation of these maps. This is the difficult research problem – automating the creation, derivation, or extraction of semantic relationships between models.

Given the links between the JAM and the LIMs, the engineering model for the Joint Action can be built. This model uses the interfaces identified in the engineering models of the participating tools to achieve the goal described by the JAM. The technical detail references the appropriate message, data, and datatype(s) for equivalent terminology. (See Figure 4)

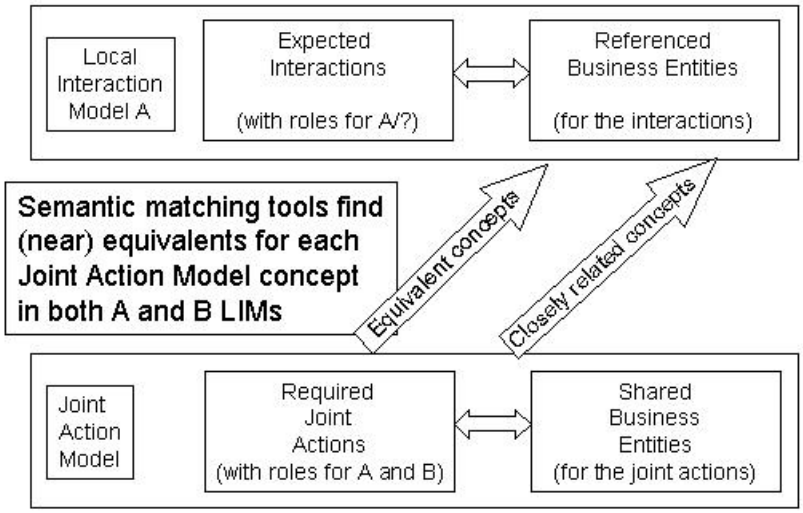

Figure 4 - Mapping Equivalents Between the JAM and the LIM

3.3 *Connector Transformation*

Given a sufficiently detailed semantic mapping, it is theoretically possible to build a tool that generates translations corresponding to the mappings. This is shown in figure 5. To achieve arbitrary transformations of syntax, structure and interactions to the lowest levels of abstraction requires that all the information be formalized. Additionally, knowledge bases for the middleware technologies must be developed, a toolkit of relevant software components must be provided, and, a characterization of that toolkit must also be formalized.

Generation of message converters is then reduced to a search problem: find the composition of available components that can transform the input available into the desired output. Although by no means necessary, optional optimization would also help. There are two significantly different engineering problems here: Conversions of messages and message elements, and dealing with differences in the actual

communications protocols. More details on all of these issues can be found in (Barkmeyer, 2004)

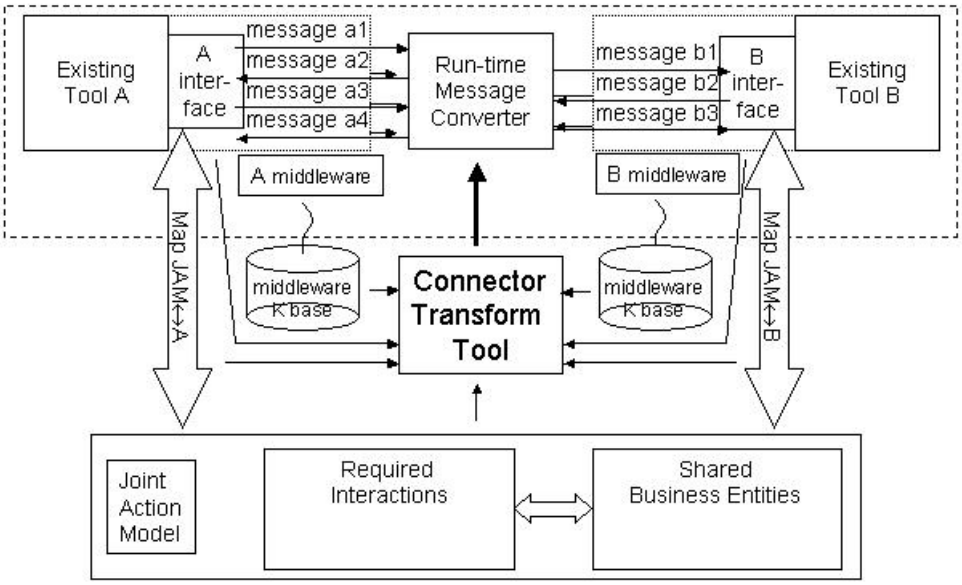

Figure 5 - Building the Run-time Message Converter

4 B2B INTEROPERABILITY TESTBED

Complementing the tools and methods emerging from the AMIS project, the objective of the industry-driven B2B Interoperability Testbed is an open, on-going testing capability to provide on-demand interoperability demonstration and testing for four stakeholders: software vendors, manufacturing organizations, standards organizations, and government.

4.1 *Motivation*

Automotive and aerospace manufacturers and the software vendors that serve them face two major interoperability barriers in their push to expand global markets. First, in the United States, multiple, competing B2B standards are emerging that lead to incompatible e-business solutions. Second, the adoption of different, possibly regional, B2B standards in places like Europe and Korea will lead to multiple e-business requirements for U.S.

companies. Removing these barriers is complicated by the nature of interoperability, which is best understood in terms of the interoperability stack shown above. This stack contains many layers each of which contain multiple standards and multiple technologies for implementing those standards. For two solutions to interoperate they must agree on both the standards and the technologies or they must develop mappings between the different choices.

Different organizations are busy creating the standards and vendors are busy using the latest technologies to implement these standards in their products. Before manufacturers buy these products, they want to be sure that (1) the standards are implemented correctly and (2) they can use the products to do business with their partners around the world. This requires substantial demonstrations that involve both users and vendors. Users provide the manufacturing scenarios and test data; vendors modify their products to implement the scenarios with the data.

Numerous such demonstrations were held at great cost and, unfortunately, great waste. This waste occurred because the software infrastructure was recreated and procedural rules reinvented for each demonstration. A number of users and vendors suggested that NIST create a persistent environment, tools, and test suites for B2B demonstration and testing. In response, NIST spawned the B2B Interoperability Testbed.

4.2 *Approach*

The testing approach adopted in the B2B Interoperability Testbed is shown in Figure 6. The OEM and the supplier represent virtual trading partners from the automotive

industry, exchanging messages using two different vendor products. NIST and testbed partners from the U.S., Korea, and Europe are developing the four tools shown in the figure: the Reflector, the Process Checker, the Content Checker, and the Syntax Checker. These tools are described briefly below; more information about these tools and the project can be found at http://www.mel.nist.gov/msid/b2btestbed/.

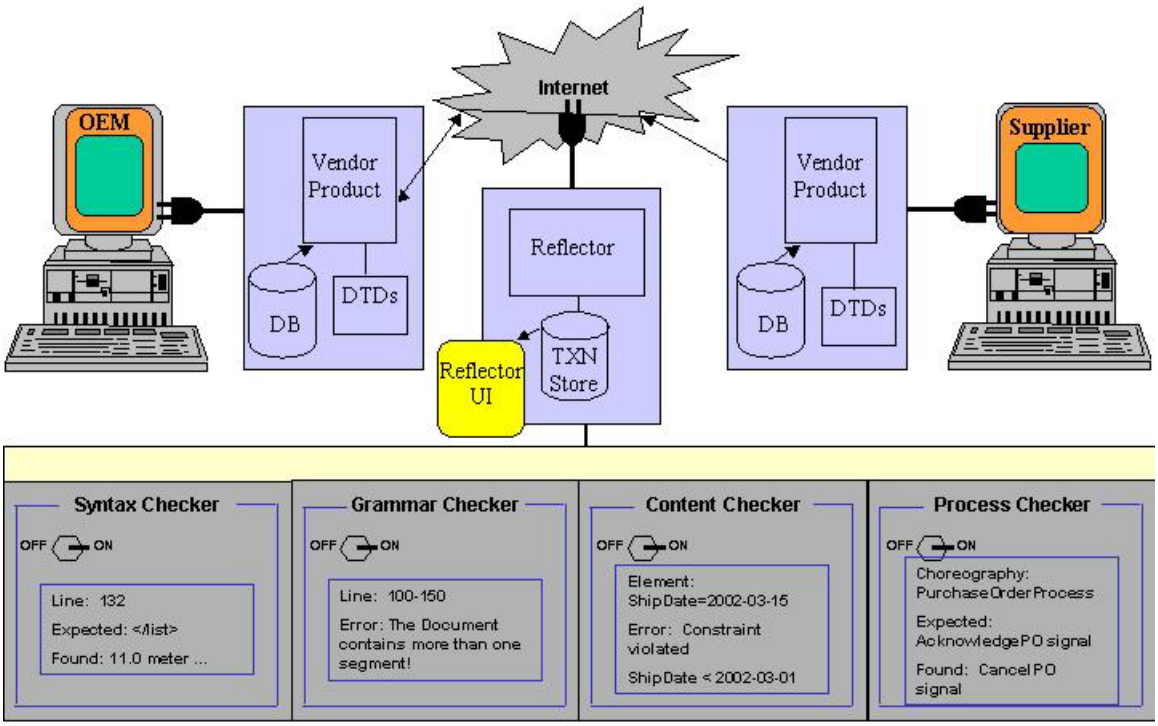

Figure 6 - B2B Testbed Tools

The Reflector is a testing tool that supports both disconnected and connected testing scenarios while allowing for the transactions to be routed to the specified end points, reflected to the originator, and stored in a permanent transaction log.

The Process Checker enables monitoring and conformance checking for choreographed transactions between business partners. The monitor currently supports ebXML BPSS

and CPA standards. The tool provides a Web-based graphical user interface to monitor in real time the business interactions based on the ebXML BPSS specification. The monitoring tool takes the ebXML BPSS and CPA instances as input and produces a graphical presentation of the collaboration as an output. The monitoring tool checks whether each message has the right sender and receiver and that they come in the right order. Further, each transaction may have a time constraint associated for its execution. Should the constraints be exceeded, the monitoring tool raises a flag that the collaboration has failed.

The Content Checker enables specification and execution of content constraints that define valid syntax, structure, or semantics of the business messages. This facility allows standard developers, users, and implementers to precisely specify, extend, and test for conformance with the semantics of a common data dictionary (lexicon). The content tool allows a user to create a profile and specify test cases using Schematron expressions and using a Web-based interface. The test cases are then stored in a customer's repository. Then another user may select and execute the test case associated with a selected customer by posting an XML document.

The Syntax Checker may be thought of as a validating XML parser whose role is to check whether a message (typically, an XML Business Document) is well-formed as specified by a standard (such as the W3C XML Schema) and that all necessary elements are present and in the right order as specified in the XML Schema instance for that business document. Additional checks may be performed by the parser (e.g., cardinality constraints) but they typically fall into the general constraints on the document structure

and component types that can be easily abstracted from the application domain and business context of usage.

The Grammar Checker may be thought of as a superset of the Syntax Checker responsible for enforcing business document structural rules that are defined in some application domain and business context, can be expressed through grammatical rules of composition and structure generation, but are not general and cannot be easily abstracted from the application context. These rules are not easily expressible in the form of XML Schema instances and require additional expressive capability, such as with the Schematron rule language. The Grammar Checker may be responsible for encoding and enforcing complex rules on optional, mandatory, and conditional cardinality that are present in real business contexts.

5 SUMMARY

In this paper, we discussed the emerging criticality of interoperability in the arena of Internet-based manufacturing. We argued that the traditional approach of painstakingly defining content standards for each application could no longer keep pace with industrial need in the long term. We proposed a new semantic-based approach and a vision, called self- integration, that promise to dramatically reduce the costs and difficulties involved in achieving interoperability. We then briefly described two projects at the National Institute of Standards and Technology that will provide the foundation for realizing that vision.

PRODUCT DISCLAIMER

Certain commercial software products are identified in this paper. This use does not imply approval or endorsement by NIST, nor does it imply that these products are necessarily the best available for the purpose.

REFERENCES

Asher, N. & Vieu, L. (1995) Toward a geometry of common sense: a semantics and a complete axiomatization of mereotopology, *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, 846-852, 1995. Montreal, Canada.

Barkmeyer, E., Feeney, A., Denno, P., Flater, D., Libes, D., Steves, M., & Wallace, E. (2004) The AMIS approach to systems integration: An overview, *NISTIR 7101*, National Institute of Standards and Technology, Gaithersburg, MD, 20899.

Gruninger, M. and Menzel, C. (2003) Process Specification Language: Principles and Applications, *AI Magazine*, 24:63-74

ISO 1994a, ISO 10303-1 (1994), *Industrial automation systems and integration — Product data representation and exchange — Part 1: Overview*, International Organization for Standardization, Geneva, Switzerland.

ISO 1994b, ISO 10303-1 (1994), *Industrial automation systems and integration — Product data representation and exchange — Part 203: Application Protocol: Configuration controlled design.*

NIST (1999), Interoperability Cost Analysis of the U.S. Automotive Supply Chain, (Planning Report #99-1), 1999, available at http://www.nist.gov/director/prog-ofc/report99-1.pdf.

NIST (2002) *Economic Impact Assessment of the International Standard for the Exchange of Product Model Data (STEP) in Transportation Equipment Industries*, (Planning Report #02-5), 2002, available at http://www.nist.gov/director/prog-ofc/report02-5.pdf